PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
H04L 29/06

A2

(11) International Publication Number: WO 97/41674

(43) International Publication Date: 6 November 1997 (06.11.97)

(21) International Application Number: PCT/US97/07271

(22) International Filing Date: 29 April 1997 (29.04.97)

(30) Priority Data:
08/641,399 30 April 1996 (30.04.96) US

(71) Applicant: 3COM CORPORATION [US/US]; 5400 Bayfront Plaza, Santa Clara, CA 95052-8145 (US).

(72) Inventors: BINDER, James, Stuart; 1277 Elkwood Drive, Milpitas, CA 95035 (US). SIDENBLAD, Paul, James; 10190 Stonydale, Cupertino, CA 95014 (US). BAKER, Roman, Gabriel; 339 Kenmore Avenue, Sunnyvale, CA 94086 (US). CONNERY, Glenn, William; 655 S. Fair Oaks #B301, Sunnyvale, CA 94086 (US).

(74) Agents: LEBLANC, Stephen, J. et al.; Townsend and Townsend and Crew L.L.P., 8th floor, Two Embarcadero Center, San Francisco, CA 94111 (US).

(81) Designated States: AU, CA, GB, JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: PACKET FILTERING BASED ON SOCKET OR APPLICATION IDENTIFICATION

(57) Abstract

A network adaptor driver includes an algorithm for determining special handling of packets by examining socket or port indications for the packets and a means for setting a special handling field in the header of outgoing packets. Specific embodiments include a table for connections that include packets without socket indications, a transmit FIFO list for determining whether a transmit FIFO is full, and a low priority receive queue. A computer readable media allowing a computer to practice the method of the invention is disclosed.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

PACKET FILTERING BASED ON SOCKET OR APPLICATION IDENTIFICATION

APPENDIX

This application is being filed with a paper appendix consisting of 95 pages containing a source code listing for an embodiment of the invention. A microfiche reproduction of this appendix will be submitted upon notice of allowance.

BACKGROUND OF THE INVENTION

This application is a Continuation-In-Part of co-assigned pending U.S. Application Serial No. 08/313,674, entitled METHOD AND APPARATUS FOR CONTROLLING LATENCY AND JITTER IN A LOCAL AREA NETWORK WHICH USES A CSMA/CD PROTOCOL.

The current invention relates to the field of electronic circuits. More particularly, the current invention relates to transmission of information between digital devices over a communications medium.

The present invention relates to the field of data communication over a computer network. A very wide variety of types of computer networks exist, each having many variations in particular implementations. The present invention will be described with reference to a particular types of computer networks, operating systems, and network interfaces. This should not be taken to limit the invention, and it will be apparent to those of skill in the art that the invention has applications in many different types of computer networks. The invention therefore should not be seen as limited except as provided in the attached claims.

Digital computer networks have become ubiquitous in academic, industry, and office environments. A number of

different aspects of computer networks are discussed in co-assigned pending U.S. applications serial nos. 08/313,674; 08/542,157; 08/506,533; and 08/329,714 each of which are incorporated herein by reference.

Most of the commonly used networks today operate in accordance with a *layered protocol suite*. In a layered protocol suite, various network tasks are divided among different pieces of hardware and software in order to allow maximum flexibility and reliability between software units.

Fig. 1 shows a simplified block diagram of a layered protocol suite as might exist between a personal computer 10 and a server computer 20 communicating via a network 15, such as two Windows-compatible type computers in an office environment running over a LAN. In a simplified form, such a system can be thought of as comprising four semi-independent layers running on each computer.

A top layer, layer 4, is the *application layer*. The application layer generally is occupied by software running on the central processor of computer 10. This software might include video conferencing software, e-mail software, the file system, or any other software that accesses network 15. The application layer 4 on computer 10 communicates data, such as a video conferencing signal, with an application layer on computer 20, by passing that data down through the protocol stack on computer 10 to the lowest layer, layer 1. Layer 1 provides a physical connection to the network and sends the data to the layer 1 device on computer 20, which then passes the data back up to its application layer.

Application layer 4 communicates data down to *protocol layer* 3. Protocol layer 3 includes software necessary to perform high level network functions such as accepting a data file from an application layer, dividing that file into packets, and attaching to those packets source and destination address information so that packets can be effectively transmitted over the network. Some examples of different protocol layer software are TCP/IP or IPX.

The protocol layer communicates with an *adapter driver layer* 2. Layer 2 generally includes software for interfacing between adapter 1 and the protocol layer 3. In prior art layered implementations, layer 2 is generally low level software whose primary purpose is to act as the interface between the protocol layer and the particular adapter hardware that is connected to or a part of computer system 10. In prior art networks, layer 2 software does not examine the contents or address of any data passing through it, but merely formats that data for the particular adapter to place on the network. Some examples of driver layer software are NDIS2 or ODI.

Adapter 1 is generally thought of as a piece of hardware for communicating signals over the network media. Adapter 1 will have different configurations depending on whether the network media is a co-axial cable, twisted-pair wire, fiber optic cable, or radio waves. The adapter may be a plug in board that connects to computer 10's system bus. Once adapter 1 receives a packet of data to be transmitted over network 15, it sends a packet over network 15 which delivers it to an adapter 1A in addressed computer 20. Once data reaches address computer 20, adapter 1A passes the packet up the layered stack first to driver layer 2A, then to protocol layer 3A, which then passes to application layer 4A.

An adapter generally includes circuitry and connectors for communication over a network medium and translates data from the digital form used by the computer circuitry a form that may be transmitted over a medium, such as a waveform. A driver is a set of instructions resident on a device that allows the device to accomplish various tasks as defined by different network protocols. Drivers are generally software programs stored on the computer in a manner that allows the drivers to be modified without modifying other hardware.

In a layered network system such as shown in Fig. 1, various networking tasks are tightly defined and performed at various network layers. In prior art systems, driver 2, for

example, receives packets of data and forwards them immediately to adapter 1 without examining the contents of the packets of data and without ever changing any of the contents of the packets of data. Driver 2 essentially exists to be an interface between a wide variety of different hardware equipment that may be provided as adapter 1 and the limited number of different network protocols 3 that might be supported by computer system 10. In prior art systems, driver 1 simply sends each packet as it receives it or sends packets in a strict FIFO order where the first packet received from protocol 3 is the first packet sent out to adapter 1 and over the network.

The layered system shown in Fig. 1 has worked well in network environments where a majority of network traffic consists of text or computer instructions which do not have particular time-critical requirements and where a computer, such as 10, only allowed one application to access the network at a time.

However, two developments in personal computer technology have shown that this system has certain limitations. The first is the wide-spread use of *socket* capable protocols and operating systems in networked environments. In a socket system, multiple applications running simultaneously on computer 10 can simultaneously send and receive data on network 15. Protocol layer 3 keeps track of for which application a particular data packet is destined by assigning different *socket numbers*. Socket numbers are requested by applications from protocol layer 3. All data packets sent and received include this number in their headers and thus can be delivered to the right application. The location of the socket identification within the packet is determined by the packet's network protocol.

The second development that has highlighted the limitations of prior art adapter drivers is that computer network traffic has increasingly begun to include real-time data such as digitized audio signals and digitized video signals, such as signals that might be used for a real time

video conferencing system. In a sockets compatible computer system, this real-time network data might be attempting to use the network at the same time that other, high volume but non-time critical, computer data is being placed on the network by a different application. Without a way for the computer system to ensure that real time video and audio data is delivered at a high priority while delays are allowed to happen in data streams that are not time critical, the real-time data can be delayed unacceptably long, hurting multi-media performance.

What is needed is a method and apparatus allowing a computer system such as 10 to send real-time data on a network at a higher priority than non-time-critical data and that requires minimal modification to other network hardware or software.

SUMMARY OF THE INVENTION

The present invention is an adapter driver for use with a network adapter card designed to recognize an identification field from an application layer program and to assign special handling or *filtering* to packets received from said application program based on the identification code. In a specific case, this filtering includes assigning a priority to individual packets. According to a specific embodiment, a driver assigns one of two priorities: a high priority (HP) or a low priority (LP). With the assignment of priority, a driver according to the present invention can manage and determine in what order packets are passed to an adapter 1 such that high priority packets are transmitted on the network before low priority packets are transmitted on the network.

According to a further embodiment of the invention, a driver according to the invention may also change data in a packet's header, thereby identifying special handling indications of said packet to other communications devices in a network.

According to a further embodiment of the invention, a driver according to the invention may detect a high priority

bit in a received packet's header and thereby deliver that packet to the layer 3 protocol ahead of other received packets.

According to a specific embodiment, a driver is designed to operate in a computer system which can function with a number of different network protocols. The driver includes instructions allowing it to read a network protocol identifier from a packet of data and then to determine the source and destination address from said packet based on the protocol used to encode said packet. An example of one type of system in which the invention may be used is the Windows(TM) operating system, developed and sold by Microsoft(TM) Corporation. In the Windows operating system, applications communicating data to a network identify packets of data with a *socket* identification. The location of this *socket* identification within the packet is determined by the packet's network protocol. (In other operating systems, the name for what Window's calls a socket is a *port*.)

In a Windows embodiment, a driver according to the invention reads a packet protocol and from the packet protocol determines the location of a socket identifier. The driver then reads the socket identifier and from it determines the priority of a packet. According to a specific embodiment, packet priority may be based on assigning particular ranges of socket identifiers to particular priorities. Such assignments may be stored in a configuration file on a computer such as computer 10 to be read by the adapter driver at startup.

A further example of an environment in which the invention may be employed is the Apple operating system. In the Apple operating system, applications also assign an identifier to every packet of data that is sent to the protocol layer.

An adapter driver according to one embodiment of the invention may include one or more priority transmit queues. These queues, according to one embodiment, are established by the driver in computer 10's main system memory. According to

a different embodiment, these queues may be placed in a dedicated memory on the adapter card. According to still another embodiment, a driver may handle high priority packets by interacting directly with memory on the adaptor card and without using priority queues. Where a driver interacts with just one priority queue, the driver includes control means allowing it to place packets at any location in the queues depending on the packet's priority. Where a driver interacts with multiple queues, a packet is located within a queue indicating that packet's priority.

According to one embodiment of the invention, an adapter driver according to the invention may be implemented entirely in software and may be designed to run on existing computer systems with existing higher layer protocol drivers and existing adapters. In this way, the advantages of the invention may be achieved by modifying only the adapter driver software on a computer such as 10, which is a relatively inexpensive investment to gain the performance benefits of the invention. This software may be embodied in computer readable executable instructions written onto a fixed medium such as a disk, for distribution to individual computer systems either via removeable or fixed disks or over a transmission medium.

In a different embodiment, the invention may be used in conjunction with other modified network software and hardware to achieve further enhancements in performance. These other modified network elements could include network switching devices that recognize a priority field set by an adapter driver according to the invention and that provide special handling for high priority packets.

In a further embodiment, a driver according to the invention is optimized to run on a modified adapter card which is designed to recognize and provide special handling for high priority packets.

Finally, the essential technique of the invention includes using an adaptor driver to examine higher layer information in a packet in order to mark the packet for

various special handling. In addition to priority handling, this special handling could include encryption, compression, or security blocking. This special handling is distinct from prior art handling of network data because it is done on a packet by packet basis by the adaptor driver layer of the protocol. The present invention is intended to encompass these variations.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a diagram illustrating a layered network protocol of a type in which the present invention may be effectively employed.

Fig. 2 is a block diagram of a computer system and an adapter with a driver according to an embodiment of the present invention.

Fig. 3 is a diagram of an example of a packet format which may be effectively handled by the present invention.

Fig. 4 is a flow chart of a method of the driver for special handling of the packets according to the present invention.

Fig. 5 is a flow chart of a method by which a driver according to the present invention sends packets to an adapter.

Fig. 6 is a flow chart of a method by which a driver according to the present invention receives and buffers low priority packets.

Fig. 7 is a block diagram of a computer system with an ability to read a computer readable medium allowing the system to operate in accordance with the invention.

Fig. 8 shows a system block diagram of computer system 10 used to execute the software of the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Fig. 2 illustrates a computer system 10 containing

an adapter 1, an adapter driver 2 according to an embodiment the invention, a protocol layer 3, and application layer 4. Driver 2 has associated with it a protocol detection algorithm (PDA) 50, a high priority transmit queue (HPTXQ) 55, and a low priority transmit queue (LPTXQ) 60. According to one embodiment, driver 2 also has associated with it an adapter FIFO transmit list 65 and a low priority receive queue (LPRXQ) 70.

Adapter 1 has associated with it an adapter transmit FIFO 110 and an adapter receive FIFO 115. Adapter 1 communicates packets through driver 2 to computer system 10 and transmits and receives packets from network 15.

Fig. 3 illustrates an example of a network protocol packet which may be effectively handled by an adapter driver according to the present invention. Fig. 3 illustrates a packet 200, containing packet header 202. Packet header 202 contains, a protocol identifier 210, a source socket identifier 204, a source address 206, a destination socket identifier 208, and a destination address 210. According to the present invention, packet header 202 may also contain a locally administered bit 212 which is used by an embodiment of the present invention to indicate special handling of the packet in the network 15.

The present invention according to one embodiment may also determine the sockets of packets where not every packet includes a socket identification, such as UDP protocol packets. In the case, a driver 2 according to the invention, maintains a table of UDP identifiers along with the socket to which a particular UDP connection is assigned.

Fig. 4 illustrates a flow chart of the operation of adapter driver 2 illustrating the method of the special handling of packets based on their source socket or their destination socket address according to an embodiment of the present invention. A packet is received by driver 2 from protocol 3 (Step S2). Driver 2, according to the invention, examines each packet it receives on transmission from protocol

3 to determine special handling needed by the packet. First, driver 1 looks for a protocol identifier in every packet it receives so as to determine which protocol created the packet (Step S4). In one embodiment this is done by a subroutine within driver 2 referred to as the priority determining algorithm (PDA) 50. Examples of possible packet protocols are PCP, EDP, UDP, or IPX.

Once PDA 50 has determined which protocol formatted a packet, PDA 50 then knows how and from where to read the header of the packet and to determine the source and destination socket numbers for the packet (Step S6).

After determining the source socket number for a packet, PDA 50 determines what special handling instructions should be assigned to the packet. This is determined by PDA 50 reading from a configuration memory 52 the ranges of socket numbers that are assigned special handling instructions and determining in which range the socket number of the current packet falls (Step S8).

According to an embodiment of the current invention, driver 2 maintains at least two transmit queues of different priorities in computer system 10 such as high priority transmit queue 55 and low priority transmit queue 60. Based on the special handling determination made in Step S8 (Step S10), driver 2 places the packet in one of these two transmit queues. In the current specific example, driver 2 will place a packet in either LPTXQ 60 (Step S12) or HPTXQ 55 (Step S16), depending on the priority determination made for the packet. According to a further embodiment, driver 2 will add priority data to the packet header (Step S14) before the packet is placed in one of the two queues. This priority data may be added to either all high priority packets, all low priority packets, or all packets.

Figure 5 illustrates a further method of the invention whereby driver 2 also determines which packets will be sent to adapter 1 for transmission on network 15. According to one example, this determination is simple and is

based on whether any packets are present in HPTXQ 55. If any packets are present HPTXQ 55, driver 2 sends those packets to adapter 1 immediately (Step T14), and allows any packets within LPTXQ 60 to wait until there are no more HP packets in queue 55 remaining to be transmitted.

According to a related further embodiment of the invention, driver 2 ensures that at least some LP packets are transmitted even when there is a very high volume of HP packets being received by driver 2, referred to herein as *fairness* (Step T12). In this embodiment, driver 2 maintains a count HPC of HP packets transmitted without interruption, and when that count reaches a certain value, driver 2 sends out a packet from LPTXQ 60 to adapter 1 if one is present.

According to a further embodiment of the invention, driver 2 is enabled to effectively schedule HP packets even when an adapter such as 1 includes a large transmit FIFO 110. In order to operate effectively with prior art adapters 1, which are not aware of the scheduling of HP and LP packets, driver 2 takes responsibility of halting LP packets being transmitted to FIFO 110 when that FIFO becomes too full. Driver 2 does this to avoid the situation where FIFO 110 contains a large number of LP packets and driver 2 receives an HP priority packet to transmit. When used with prior art adapters, driver 2 has no way to force the HP packet ahead of other packets stored in FIFO 110, and the HP packet would have to wait.

A driver 2 according to this embodiment of the invention, avoids this problem by maintaining an adapter FIFO transmit list 65, in which driver 2 logs the presence of all packets sent to adapter 110. Driver 2 then examines the FIFO free space value returned from adapter 110 to determine which packets that it has stored have been sent by the adapter onto network 15. When driver 2 determines that a certain number of low priority packets reside on transmit FIFO 110 and have not been sent, driver 2 halts forwarding any further low priority packets to adapter 1 until the number of low priority packets stored in transmit FIFO 110 again falls below a certain

transmit FIFO; and

a transmit FIFO list memory for storing indications of which packets in a transmit FIFO have been sent wherein said adapter driver halts placing additional low priority packets in said transmit FIFO when a specified number of packets are determined to be residing is said transmit FIFO.

14. A network adapter driver according to claim 10 further comprising:

configuration memory for storing priority indications based on ranges of source socket identifiers of said packets.

15. A network adapter driver according to claim 10 further comprising:

a packet header modification algorithm for modifying a data field in a packet header in order to indicate a priority of said packet.

16. A network adapter driver according to claim 10 further comprising:

a table for storing socket indications for protocols that include packets that do not have socket indications in each packet header.

17. A network adapter driver according to claim 10 further comprising a first transmit queue interface for a high priority transmit queue and a said second transmit queue interface for a low priority transmit queue.

18. A network adapter driver according to claim 10 wherein said adapter driver is a software module that may be installed on a computer system with an adapter without any modifications to said computer system or said adapter and wherein said queues are allocated by said driver from said computer system's system memory.

19. A method for scheduling transmission of packets in an adapter driver comprising the steps of:

receiving a packet of data from a higher protocol layer said packet containing a packet header;

reading from said packet header and indication of the protocol that constructed that packet;

using said protocol to determine where in said header a source socket identifier is located and reading said socket identifier;

using said identifier to determine a priority for said packet; and

placing said packet into a transmit queue in a location indicated by said priority.

20. The method according to claim 19 further comprising:

sending packets out of said higher priority queue locations before sending them out of lower priority locations.

21. The method according to claim 19 further comprising:

adding data to said packet header indicating said priority before placing said packets a queue.

22. The method according to claim 19 further comprising:

adding data to said packet header indicating said priority before placing said packets a queue.

23. A fixed computer readeable medium containing computer executable program, which, when loaded into an appropriately configured computer system will cause the computer to perform the method of claim 19.

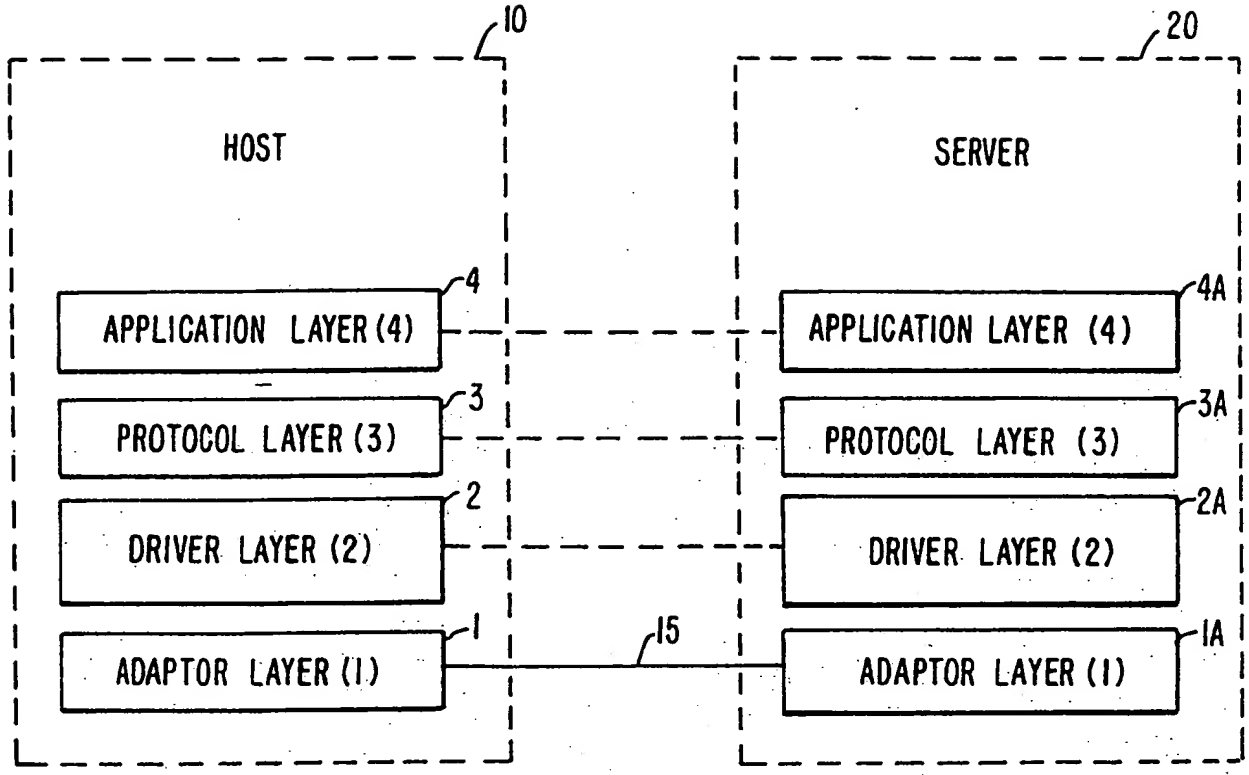
10
20
HOST
SERVER
4
APPLICATION LAYER (4)
4A
APPLICATION LAYER (4)
3
PROTOCOL LAYER (3)
3A
PROTOCOL LAYER (3)
2
DRIVER LAYER (2)
2A
DRIVER LAYER (2)
1
ADAPTOR LAYER (1)
15
1A
ADAPTOR LAYER (1)

FIG. 1.

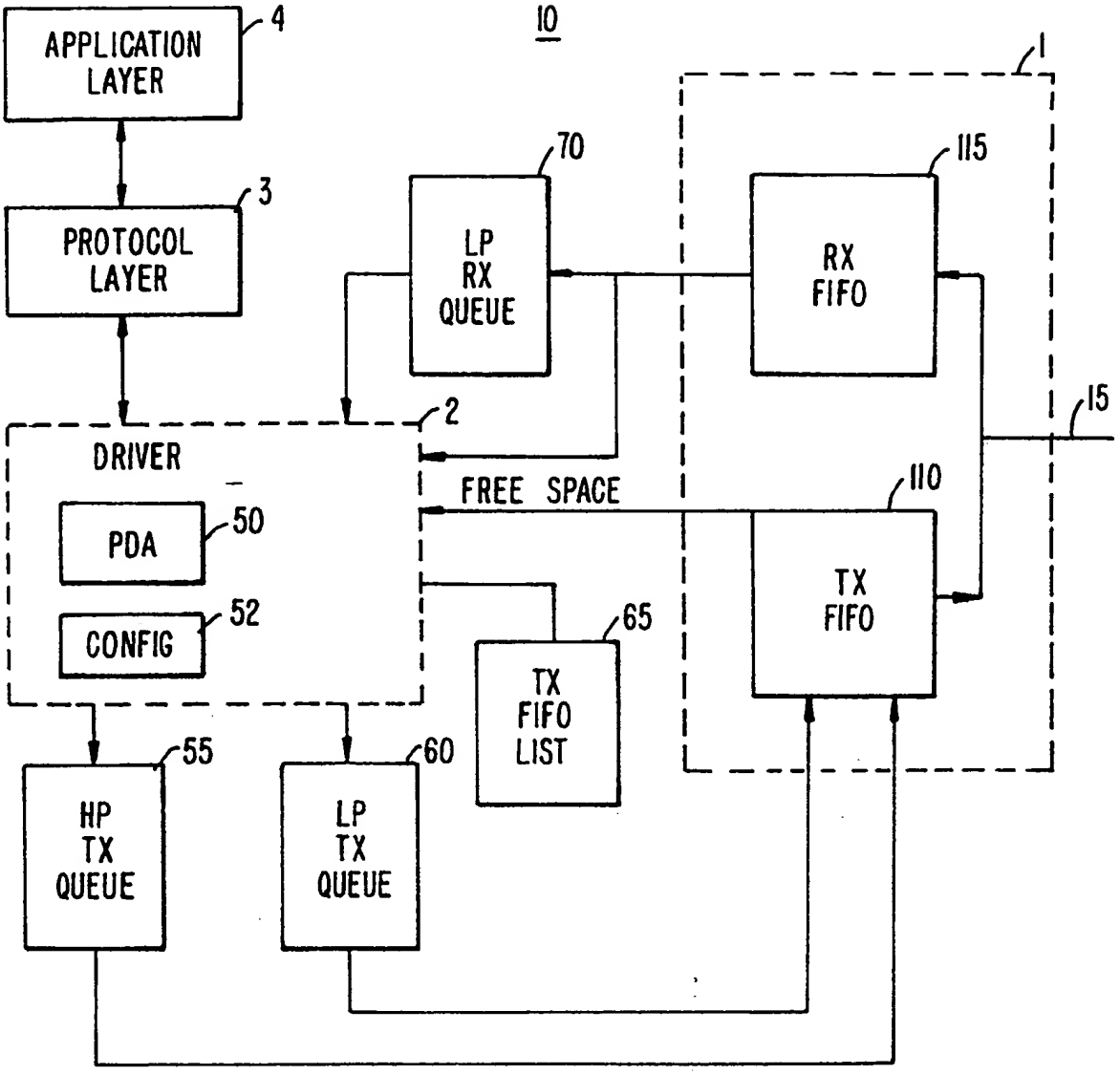
10
APPLICATION LAYER
4
PROTOCOL LAYER
3
DRIVER
2
PDA
50
CONFIG
52
HP TX QUEUE
55
LP TX QUEUE
60
LP RX QUEUE
70
TX FIFO LIST
65
FREE SPACE
RX FIFO
115
TX FIFO
110
1
15

FIG. 2.

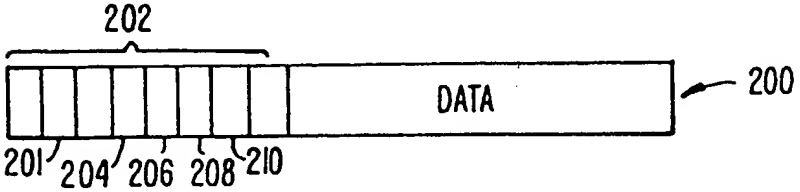
202
DATA
200
201
204
206
208
210

FIG. 3.